US006205551B1

(12) **United States Patent**
Grosse

(10) **Patent No.: US 6,205,551 B1**
(45) **Date of Patent: Mar. 20, 2001**

(54) **COMPUTER SECURITY USING VIRUS PROBING**

(75) Inventor: **Eric Grosse**, Berkeley Heights, NJ (US)

(73) Assignee: **Lucent Technologies Inc.**, Murray Hill, NJ (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/015,563**

(22) Filed: **Jan. 29, 1998**

(51) **Int. Cl.**^{7} **G06F 12/14**

(52) **U.S. Cl.** **713/201**

(58) **Field of Search** 708/135; 709/224; 713/200–201; 714/715, 716, 748

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,361,359		11/1994	Tajalli et al.	395/700
5,414,833		5/1995	Hershey et al.	395/575
5,440,723		8/1995	Arnold et al.	395/181
5,473,769		12/1995	Cozza	395/183.15
5,550,984		8/1996	Gelb	395/200.17
5,613,002		3/1997	Kephart et al.	380/4
5,623,600	*	4/1997	Ji et al.	395/187.01
5,623,601		4/1997	Vu	395/187.01
5,706,507		1/1998	Schloss	395/615
5,815,571	*	9/1998	Finley	380/4
5,832,208	*	11/1998	Chen et al.	395/187.01
5,889,943	*	3/1999	Ji et al.	395/187.01
5,948,104	*	9/1999	Gluck et al.	713/200
5,960,170	*	9/1999	Chen et al.	395/183.14
5,961,644	*	10/1999	Kurtzberg et al.	713/200
5,987,610	*	11/1999	Franczek et al.	713/200
6,009,475	*	12/1999	Shrader	709/249
6,041,041	*	3/2000	Ramanathan et al.	714/25

* cited by examiner

Primary Examiner—David A. Wiley
(74) *Attorney, Agent, or Firm*—Donald P. Dinella

(57) **ABSTRACT**

A technique for determining whether particular clients within a computer network are universally configured in accordance with the desired network security features of the computer network. A probe is randomly inserted within incoming files, e.g., at a firewall in the computer network. The probe is configured as a function of a particular execution task, e.g. a known virus, such that in a properly configured client the probe will not execute and the firewall does not detect a security breach. However, if the client is misconfigured, i.e., not in compliance with the standard network security features, the probe will execute and trigger an alarm in the firewall indicating that the client is vulnerable to a security breach. Advantageously, a network security administrator can take appropriate action to correct those clients which are misconfigured.

28 Claims, 2 Drawing Sheets

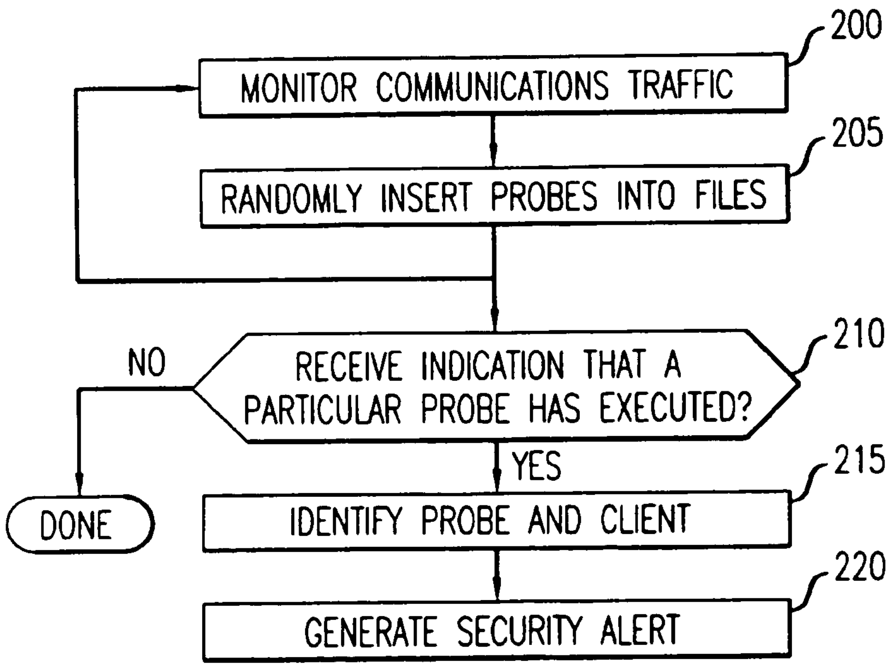

200
MONITOR COMMUNICATIONS TRAFFIC
205
RANDOMLY INSERT PROBES INTO FILES
210
RECEIVE INDICATION THAT A PARTICULAR PROBE HAS EXECUTED?
NO
YES
DONE
215
IDENTIFY PROBE AND CLIENT
220
GENERATE SECURITY ALERT

FIG. 1

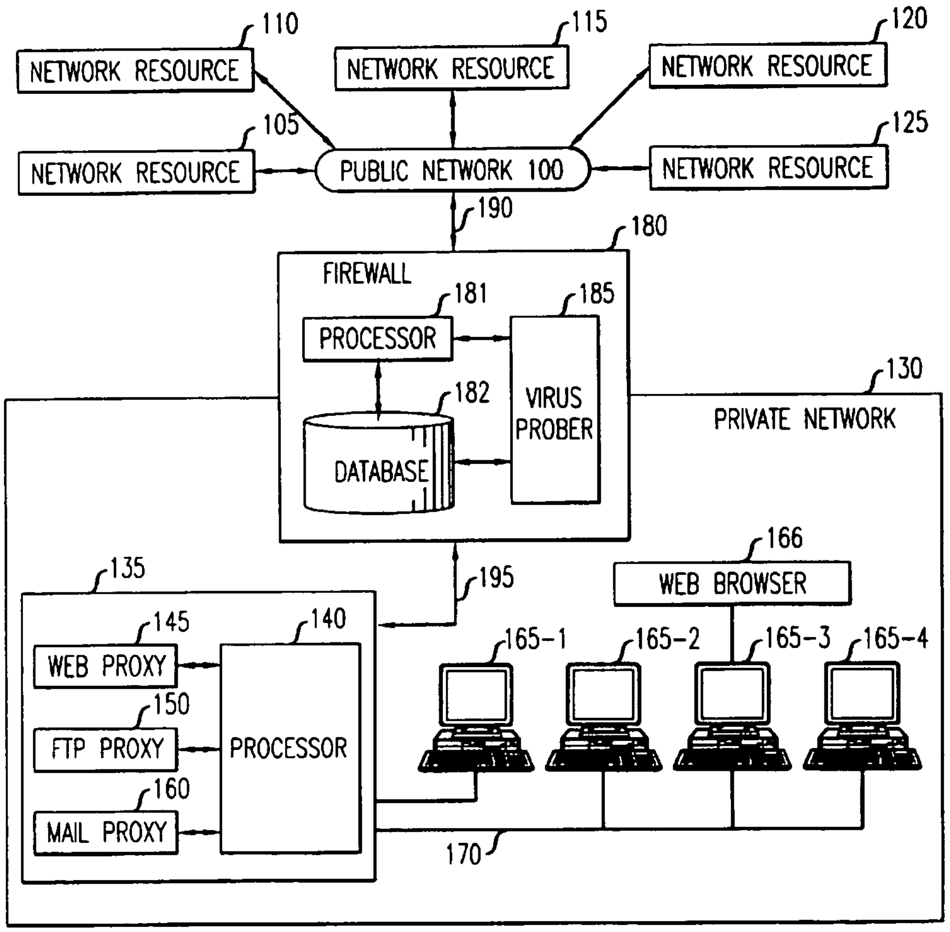

110
NETWORK RESOURCE
115
NETWORK RESOURCE
120
NETWORK RESOURCE
105
NETWORK RESOURCE
PUBLIC NETWORK 100
125
NETWORK RESOURCE
190
180
FIREWALL
181
PROCESSOR
185
VIRUS PROBER
182
DATABASE
130
PRIVATE NETWORK
195
166
WEB BROWSER
135
145
WEB PROXY
140
PROCESSOR
150
FTP PROXY
160
MAIL PROXY
165-1
165-2
165-3
165-4
170

FIG. 2

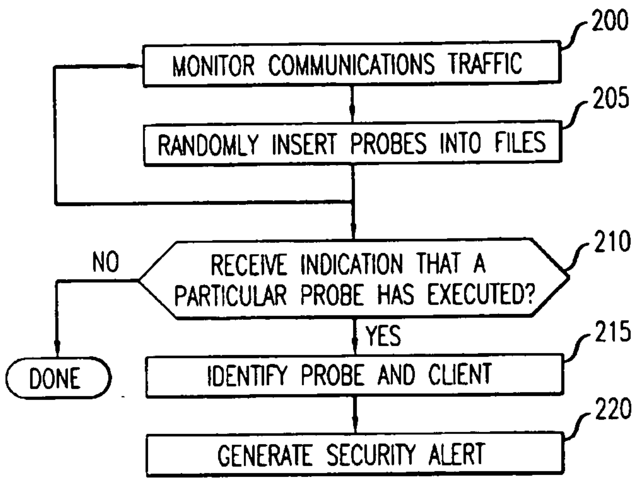
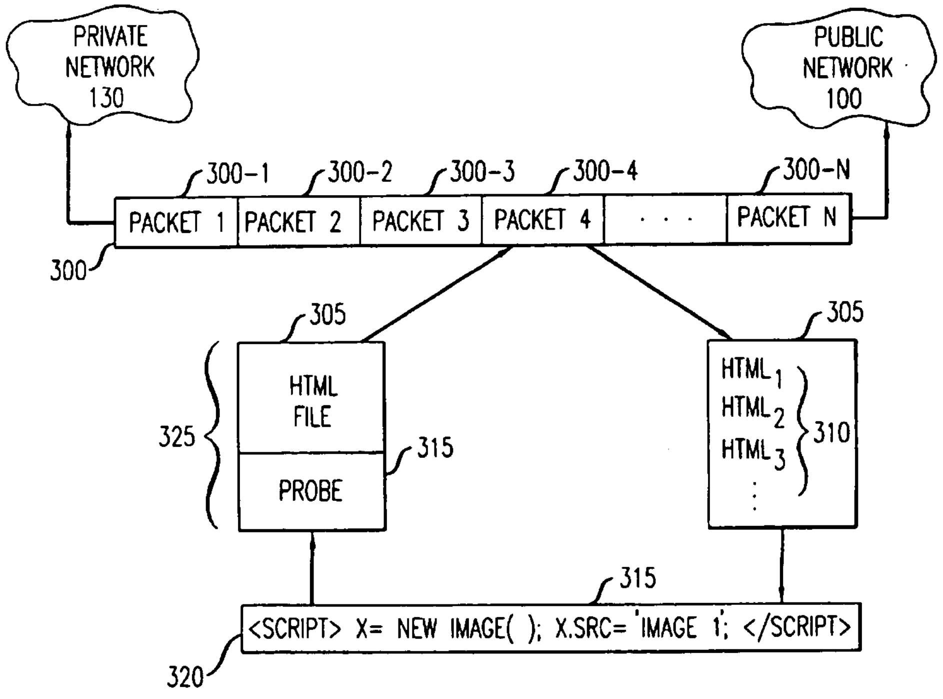

200
MONITOR COMMUNICATIONS TRAFFIC
205
RANDOMLY INSERT PROBES INTO FILES
210
RECEIVE INDICATION THAT A PARTICULAR PROBE HAS EXECUTED?
NO
DONE
YES
215
IDENTIFY PROBE AND CLIENT
220
GENERATE SECURITY ALERT

FIG. 3



06/13/2003, EAST Version: 1.04.0000

COMPUTER SECURITY USING VIRUS PROBING

FIELD OF THE INVENTION

The present invention relates to network security and, more particularly, to a technique for the verification of security measures employed in computer networks.

BACKGROUND OF THE INVENTION

Advances in communications technology and the availability of powerful desktop computer hardware has increased the use of computers to access a variety of publicly available computer networks. Today, a tremendous amount of information is exchanged between individual users located around the world via public computer networks, e.g., the Internet. One class of users includes private individuals and professional users interconnected via a private network, e.g., a corporate intranet. The exchange of information between private and public computer networks has presented a variety of critical security issues for the protection of information on the private computer networks and the overall functionality of the private computer network itself.

Computer network security, at a minimum, is directed to ensuring the reliable operation of computing and networking resources, and protecting information within the network from unauthorized disclosure or access. Various security threats exist which pose increasingly difficult challenges to such network security. In particular, some of the most sophisticated types of security threats are posed by programs which exploit certain vulnerabilities within network computing systems. To name a few, these program-related security threats include well-known logic bombs, trapdoors, trojan horses, viruses and worms, as described, e.g., by W. Stallings, *Network and Internetwork Security Principles and Practice,* Prentice-Hall, Inc., Englewood Cliffs, N.J., 1995. Such well-known software program threats either work independently (e.g., worms) to achieve their desired security breach, or require the invocation of a host program to be invoked to perform the desired disruptive actions (e.g., trapdoors, logic bombs, trojan horses or viruses.) Indeed, there are numerous well publicized accounts of such programs being used to improperly breach the security of private computer networks and cause severe damage (see, e.g., J. Hruska, *Computer Viruses and Anti-Virus Warfare,* Second edition, Ellis Horwood Limited, New York, 1992.) Such damage has included the destruction of electronic files, alteration of databases, or the disabling of the computer network itself or computer hardware connected to the affected network.

Network administrators responsible for the operation of private computer networks employ a variety of security measures to protect the network from external security breaches such as the introduction of computer viruses. One technique uses so-called firewalls. This security scheme essentially places a separate computer system, i.e., the firewall, between the private network and the public network, e.g., the Internet. These the firewalls are software-based gateways that are typically installed to protect computers on a local area network ("LAN") from attacks by outsiders, i.e., unauthorized users. The firewall maintains control over communications from and to the private network. Essentially, the firewall imposes certain security measures on all users employing the private network. For example, firewalls may block access to new Internet services or sites on the World Wide Web ("WWW") because the security consequences are unknown or not accounted for by

the present firewall configuration. One potential installation configuration of a firewall is that WWW clients can no longer directly contact WWW servers. Typically, this proves too restrictive, and network administrators employ so-called "proxy servers". Proxy servers are designed with certain features which provide for the forwarding of requests from WWW clients through the firewall thereby providing communication flow to and from servers on the Internet.

Recently, firewall vendors have included so-called "virus filtering" features to address critical security issues associated with virus infection. More particularly, this virus filtering at the firewall is conceptually similar to well-known virus scanning typically employed on client machines, e.g., personal computers, which reside within a LAN in a conventional client/server arrangement. In such client-based virus detection, virus scanning is accomplished using a program which searches through, e.g., the operating system, executable files, system files, boot records, and memory, of the client looking for the presence of undesirable software entities. Computer viruses are detected by the virus scanner by using previously defined "virus signatures" associated with each virus. The virus signature is typically a fixed-length signature pattern, e.g., a 16 to 24 byte pattern, extracted from the known virus by the vendor of the virus scanning software. The virus scanning software contains a list of signatures for known computer viruses and scans the various files in a particular client looking for a match to a particular virus signature. If a match is found, this entity of the client is "infected" and the user is notified accordingly.

The incorporation of virus filtering within commercially available firewalls provides for virus detection by scanning files transmitted through the firewall. While this provides the firewall with additional network security capabilities, implementing the virus filter at the firewall presents certain operational difficulties which include: (1) a substantial amount of processing must be accomplished at the firewall which degrades network performance through the introduction of latency which affects applications executing in the network; and (2) the firewall itself contains less operational and data intelligence with regard to individual clients in the network which leads to a less precise scan of the incoming data by the firewall as could be accomplished by a client-based virus scanner.

Therefore, given the potential drawbacks in firewall-based virus filtering, most network security administrators opt for providing virus screening in the client machines across the network rather than in the firewall itself. Currently, a number of popular commercial computer virus scanners are used for such client-based scanning. Typically, network security administrators will select a particular commercially available virus scanning program and install the program across all the clients of the network. Of course, the effectiveness of the virus scanning software is as function of the uniformity of installation and periodically updating the virus signature listing used by the software to included newly identified viruses. As will be appreciated, for very large client/server networks the task of ensuring that the virus detection software is universally installed and updated on all clients is significant and not always achievable. A client-by-client inspection is labor intensive and cannot be undertaken on a frequent enough basis to ensure conformity. Therefore, individual users are typically responsible for updating their virus scanning software by, e.g., downloading the most current virus signature listing from a central source. Of course, the lack of diligence and infrequency of such updates by individual users can lead to potential secure breaches within the network.

A need exists therefore for ensuring that network security features are universally configured throughout a computer network.

SUMMARY OF THE INVENTION

The present invention provides a technique for determining whether particular clients within a computer network are universally configured in accordance with the desired security features of the computer network. In accordance with the invention, a probe is randomly inserted within incoming files in the computer network. Illustratively, the insertion of probes occurs in a firewall which separates the computer network from other networks. The probe, in accordance with an embodiment of the invention, is configured as a function of a particular execution task, e.g. a known virus, such that in a properly configured client the probe will not execute and the firewall does not detect a security breach. However, if the client is misconfigured, i.e., is not in compliance with the standard network security measures, the probe will execute and trigger a security alert in the firewall indicating that the client is vulnerable to a security breach. Advantageously, a network security administrator can take appropriate action to correct those clients which are misconfigured.

In preferred embodiments of the invention, the probe is configured as a virus probe in the form of a trojan horse which, if executed, on a client will launch a signal back to the firewall indicating that the client is misconfigured. In further embodiments of the invention, the signal back to the firewall is a User Datagram Protocol ("UDP") packet. In accordance, with a further embodiment of the invention, the virus probe is inserted upon a first Internet access from a particular IP address or browser type, and thereafter virus probes are inserted at random intervals.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 shows an exemplary system embodying the principles of the invention;

FIG. 2 is a flowchart of operations illustratively performed by the firewall of FIG. 1 in implementing the present invention, and

FIG. 3 show an illustrative communications traffic stream transmitted in the system of FIG. 1 and configured in accordance with the invention.

DETAILED DESCRIPTION

The present invention provides a technique for determining whether particular clients within a computer network are universally configured in accordance with the desired security features of the computer network. In accordance with the invention, a probe is randomly inserted within incoming files, illustratively, at a firewall in the computer network. The probe, in accordance with an embodiment of the invention, is configured as a function of a particular execution task, e.g. a known virus, such that in a properly configured client the probe will not execute and the firewall does not detect a security breach. However, if the client is misconfigured, i.e., is not in compliance with the standard network security measures, the probe will execute and trigger an alarm in the firewall indicating that the client is vulnerable to a security breach. Advantageously, a network security administrator can take appropriate action to correct those clients which are misconfigured.

FIG. 1 shows an exemplary system embodying the principles of the invention. As shown in FIG. 1, the system includes public network 100, e.g., the Internet, and network

resources 105, 110, 115, 120 and 125. Illustratively, network resources 105 through 125 can be linked together using files written in the well-known Hypertext Mark-up Language ("HTML") thereby representing the well-known WWW. The WWW and HTML are described in more detail, e.g., by B. White, *HTML and the Art of Authoring for the World Wide Web*, Kluwer Academic Publishers, Norwell, Mass; 1996. Illustratively, private network 130 is a network located within a particular user site, e.g., a corporation's headquarters building, having user terminals 165-1, 165-2, 165-3 and 165-4 linked together via LAN 170. As will be appreciated, user terminals 165-1 through 165-4 can be stand-alone personal computers or network terminals. For simplicity of explanation herein, only one such LAN configuration is shown in FIG. 1, however, as will be appreciated private network 130 may include several such LAN configurations similar in nature to LAN 170. A particular user of any one of user terminals 165-1 through 165-4 may cause a client program executing on, e.g., user terminal 165-3, to request certain resources which are available on the WWW, e.g., network resources 105–125. As mentioned previously, such requests to the WWW via the Internet from private network 130 pose certain security risks to both private network 130 and user terminals 165-1 through 165-4. Thus, as shown in FIG. 1, private network 130 includes firewall 180 and proxy server 135 which are configured to delivery certain security features, in accordance with the invention, to protect private network 130 and its various computing resources.

As discussed previously, network administrators responsible for the operation of private computer networks, e.g., private network 130, employ a variety of security measures to protect the network from external security breaches such as the introduction of computer viruses. One technique places a separate computer system, i.e., the firewall, between the private network and the public network, e.g., the Internet. The firewall monitors and maintains control over communications from and to the private network. More particularly, where a private network employs a firewall, the firewall first determines if the requested connection between a user terminal in the private network and the public network is authorized. The firewall serves as an intermediary between the user terminal in the private network and the public network and, if the connection is authorized, facilitates the requisite connection between the two networks. Alternatively, if the connection is unauthorized, the firewall prevents any connection between the networks from occurring.

In accordance with the illustrative embodiment of the invention shown in FIG. 1, proxy server 135 includes processor 140, web proxy 145, file transport protocol ("FTP") proxy 150 and mail proxy 160. As will be appreciated, these illustrative proxies enable the proxy server, working in conjunction with the firewall, to provide security features for WWW/Internet access, file transfers and electronic mail, respectively. For example, web proxy 145 is used when a user desires to access particular "web pages" on the WWW from private network 130. Illustratively, a user employing user terminal 165-3 may access certain web pages on the WWW using web browser 166. Web browsers are well-known software application programs (e.g., Netscape® v. 5.0, available from Netscape Communications) which enable a user to traverse the WWW and access the vast amount of information available throughout the WWW. Thus, web browser 166 receives an input request from the user of user terminal 165-3 and attempts to locate the information on the WWW by establishing a connection with the appropriate resource, e.g., network resource **105**, on the WWW through public network **100**. The connection between user terminal **165-3** and network resource **105** is established using proxy server **135**, web proxy **145** and firewall **180**. More particularly, web proxy **145**, acting on behalf of web browser **166**, will attempt to establish a conventional Transfer Control Protocol/Internet Protocol ("TCP/IP") connection between user terminal **165-3** and network resource **105**. As is well-known, TCP/IP is the protocol which is used in describing the way in which information is transferred across the Internet. Essentially, TCP/IP separates information into individual packets and routes these packets between the sending computer, e.g., server, and the receiving computer, e.g., client. TCP/IP and Internet communications are discussed in more detail, e.g., by D. Comer., *Internetworking with TCP/IP*, Third edition, Prentice-Hall, Englewood Cliffs, N.J., 1995. In the present embodiment, the TCP/IP connection between user terminal **165-3** and network resource **105** is made across communication channels **190** and **195**, respectively, which establish connection between public network **100**, private network **130** and, ultimately, user terminal **165-3**.

As seen from FIG. **1**, all communications traffic between public network **100** and private network **130** necessarily passes through firewall **180**. In recognition of this communications traffic attribute, I have realized that the firewall **180** provides a preferred location for implementing the security advantages of my invention. Illustratively, in accordance with the preferred embodiment of the invention, firewall **180** illustratively includes processor **181**, database **182**, and virus prober **185** which randomly inserts probes within incoming files from, e.g., public network **100**, to, e.g., private network **130**. In accordance with the invention, the probes inserted by the virus prober **185** are individual programs which will trigger particular actions upon execution. In accordance with an embodiment of the invention, the probe is a virus probe configured as a trojan horse which, if executed, on a client will launch a signal back to the firewall indicating that the client is misconfigured. Typically, from a computer virus perspective, a trojan horse is a secret, undocumented entry point placed into a useful application program by an unauthorized user, e.g., computer hacker. In the normal course of execution of the useful application program by a user the trojan horse is also executed thereby launching the undesired actions. Trojan horses are described in more detail in Stallings, supra. at pp. 238–241. For example, a trojan horse can be created to gain access to the files of another user on a shared computer system, wherein the unauthorized user creates a trojan horse program that, when executed, changes the authorized user's file permissions so that their files become readable by any user. This embodiment of the invention utilizes particular features of the trojan horse for delivery of various security advantages to computer networks as discussed in more detail below.

In accordance with the invention, the virus probe inserted by virus prober **185** at firewall **180** is benign in that the probe is designed to provide a signal back to firewall **180** if executed, rather than perform some destructive action as in the conventional trojan horse sense as described previously. The security features of the invention are preferably implemented and realized at the firewall, e.g., firewall **180**, because in networks where firewalls are employed all communications traffic must pass through the firewall. Thus, the firewall is an ideal location for inserting probes in accordance with the invention. However, will be appreciated, the principles of the invention are also realized in other network environments and configurations. For example, in accordance with a further embodiment of the invention, the insertion of probes can be accomplished using a particular

proxy server within a network that is known to have a high rate of common access and is trusted. For example, a trusted server within a private network which mainly provides an online telephone directory is also an excellent candidate for implementing the principles of the invention due to the fact that this server will be utilized by a high number of user within the private network. Thus, the security features delivered by the present invention are realized in a variety of network, hardware and software configurations including, but not limited to, the system configuration of FIG. **1**.

The operations of delivering network security through the insertion, monitoring and execution of probes in accordance with the invention is shown in the illustrative operations of FIG. **2**. In accordance with the preferred embodiment of the invention, as described above, the operations of FIG. **2** are initiated within firewall **180**. More particularly, in accordance with the invention, the communications traffic stream, e.g., in and out of private network **130**, is continually monitored (block **200**.) During the course of monitoring the communications traffic stream transmitted across the network, probes are randomly inserted into incoming files (block **205**) destined for private network **130**. The structural aspects of the probe of the invention are described below in more detail with regard to FIG. **3**. In accordance with the invention, the probes are designed, if executed on a client, to trigger a signal indicative of a security alert.

Illustratively, the signal can be a request for a network resource. Since all such requests must be made through the firewall, this ensures that when a probe configured in accordance with the invention triggers such a request, the request can effectively be utilized as the signal to the firewall. That is, such signals triggered by the probe will be immediately recognizable by the firewall. In further embodiments of the invention, the signal can be in the form of a conventional User Datagram Protocol ("UDP") packet. As will be appreciated, UDP is a transport protocol which runs on top of the conventional TCP/IP protocol and provides a low overhead mechanism for two applications to quickly exchange small amounts of data. UDP requires less overhead than typical TCP/IP packet exchanges because UDP is a less secure protocol than TCP/IP. That is, UDP is transaction oriented, and packets may be duplicated, lost or received in a different order than as originally sent. In contrast, TCP/IP is more reliable because the protocol goes to significant lengths (e.g., generating checksums, acknowledging the receipt of packets, retransmitting lost packets) to insure that data arrives at its destination intact. Since UDP has no such overhead it is considerably faster than TCP/IP and is ideal for applications, as in various embodiments of the invention, that transmit short bursts of data, need faster network throughput, or do not require verification of delivery at the destination. As will be appreciated, other types of signal configurations, in addition to those described above, which will be equally effective in delivering the various aspects of the invention.

Thus, when firewall **180** receives the security alert indication, e.g., UDP packet, that a particular probe has executed (block **210**), the firewall will identify the probe and client (block **215**) and generate the security alert (block **220**.) The nature and type of the security alert generated, in accordance with the invention, can be in a variety of forms. Illustratively, the security alert generated by firewall **180** could be an immediate notification to the network administrator indicating that a particular client or clients within the network currently present a security risk. In a further embodiment of the invention, as probes are executed by various ones of the clients within the network, a log entry is made in a master file, e.g. stored in database **182**, which can be accessed by the network administrator at regular intervals or a printed report could be generated from the log for review by the administrator.

Advantageously, the invention provides a technique for determining whether particular clients with a computer network are universally configured in accordance with the desired network security features of the computer network. For example, one conventional security measure dictated by most network administrators is a policy that all users within a network, e.g., private network **130**, disable certain features of their web browser software, e.g. Netscape®, and in particular the Javascript interpreter feature of the web browser. Javascript is described in more detail, e.g., by D. Flanagan, *Javascript The Definitive Guide,* Second edition, O'Reilly & Associates, Sebastopol, Calif., 1997. Briefly, Javascript is a well-known interpreted programing language useful, e,g., in developing programs which relate to and involve web browsers and HTML. For example, when a web browser includes a Javascript interpreter, the browser enables executable content, e.g., programs, to be distributed over the Internet (and WWW) in the form of Javascript "scripts". When the script is loaded into a Javascript-enabled browser the script is executable and will produce particular output as defined by the Javascript instructions of the script. Thus, Javascript allows for the control over the web browser, and also the content of that which appears in a web page, e.g., HTML forms. As will be appreciated, these features which are enabled through the use of Javascript present serious network security risks.

The import of the present invention in the web browser environment described above is detailed in the following illustrative embodiments. Turning our attention to FIGS. **1** and **3**, private network **130** includes a plurality of users employing user terminals **165-1** through **165-4**. As discussed previously, each user terminal can be configured with a web browser such as web browser **166** executing on user terminal **165-3**. As will be readily understood, the configuration of user terminal **165-3** is easily replicated on each of the other user terminals within the private network but for purposes of clarity herein only one such configuration is shown in FIG. **1**. Thus, in conformance with the security policy for private network **130**, all web browsers are to have their Javascript interpreter disabled to prevent the execution of scripts which may be introduced from foreign sources, e.g., a public network, and subject the private network to various security risks. Of course, such a security measure is only effective if the users of the network comply. Typically, in most private networks there will exist, at any one time, particular user terminals which are not in compliance with the prescribed security measures. Thus, these non-complying user terminals represent a security risk to the entire network and a constant challenge to the network administrator for insuring full compliance with all security measures across the entire private network.

As discussed previously, the invention provides a technique for determining whether particular clients with a computer network are universally configured in accordance with the desired network security features of the computer network. More particularly, firewall **180** is configured, as described above, in accordance with the invention to insert probes into the incoming communications traffic stream to private network **130**. FIG. **3** shows an illustrative incoming communications traffic stream **300** and the insertion of an illustrative probe in accordance with the principles of the invention. In particular, communications traffic stream **300** includes a series of individual packets **300-1** through **300-n**, e.g., TCP/IP packets, carrying data from public network **100** to private network **130**. Thus, in accordance with the invention, firewall **180** monitors communication traffic stream **300** and randomly inserts probes into incoming files within particular ones of the packets. For example, packet **300-4** contains incoming file **305**, illustratively a file having a series of HTML instructions **310**. In accordance with the invention, virus prober **185** inserts probe **315**, illustratively, at the end of HTML instructions **310**. In accordance with various embodiments of the invention, probe **315** is inserted upon a first Internet access from a particular IP address (i.e., client) or browser type, and thereafter virus probes are inserted at random intervals. Illustratively, probe **315** is a virus probe in trojan horse form, as previously discussed, wherein the insertion of probe **315** into file **305** results in edited file **325**. Thereafter, edited file **325** proceeds in the transmission of communications traffic stream **300** to private network **130**.

Illustratively, probe **315** is a single Javascript instruction **320**. As shown, Javascript instruction **320** is of the form "<SCRIPT>x=new image( ); x.src='image1';</SCRIPT> which, as discussed above, is an interpreted scripting language statement for controlling a web browser. Further, illustratively, "image1" is a unique string of characters for identifying probe **315**. Basically, probe **315** is a trojan horse which directs the web browser to allocate an off-screen bitmap space, i.e., "new image( )" and download a small image, i.e., "image1". In accordance with various embodiments of the invention, the probes can either be stored in database **182** for access by virus prober **185** or stored locally within virus prober **185** itself In accordance with a further embodiment, probes can be downloaded by network administrators from a central source, e.g., the Internet, and added to the existing probe library. In accordance with the invention, if web browser **166** is in compliance with the illustrative network security feature which requires that all web browsers have their Javascript interpreter disabled, probe **315** will not execute and firewall **180** will not generate any security alert. However, in accordance with the invention if web browser **166** is misconfigured, probe **315** will execute causing web browser **166** to initiate a request for the image file, i.e., image1. As described previously, the mere request by web browser **166**, in accordance with an embodiment of the invention, for a network resource is captured by firewall **180** thereby serving as the signal of a security alert. There is no reason for a properly configured web browser to ask for such a network resource, i.e., image1, unless it is improperly configured and outside of established network security measures. That is, execution of probe **315** means that web browser **166** is Javascript enabled which is not in compliance with the desired security measure of the private network **130** and therefore poses a security risk to the network.

As described previously, a further embodiment of the invention employs a UDP packet as the signal back to the firewall when a security alert has occurred. In such an embodiment, file **305** is, illustratively, a file containing certain executable instructions. As is well-known, files having the extension ".exe" are binary executable files. Thus, in accordance with the invention, probe **315** will be inserted into file **305** at an appropriate location where it is known to be safe for overwriting a small number of bytes of file **305** for insertion of probe **315**. In accordance with this embodiment of the invention, probe **315** will launch a UDP packet when a security alert occurs. Illustratively, the actual machine instructions inserted into file **305** are generated using, i.e. compiling, the following code segment written in the well-known C programming language:

```
struct sockaddr_in sin={0,9,{0xF14E8A11},0,0,0,0,0,0,0,
  0};
int s=socket(PF_INET,SOCK_DGRAM,0);
connect(s,&sin,sizeof(sin));
write(s,0x88,1);
close(s);
```

As will be appreciated by those skilled in the art, the above illustrative C program segment, after being compiled into machine code, is inserted as probe **315** into file **315** and will generate the desired UDP packet upon probe execution. That is, if probe **315** is executed on a particular user terminal, a UDP packet will be launched to firewall **180** as the signal indicating that the user terminal is a potential security risk.

The foregoing merely illustrates the principles of the present invention. Therefore, the invention in its broader aspects is not limited to the specific details shown and described herein. Those skilled in the art will be able to devise numerous arrangements which, although not explicitly shown or described herein, embody those principles and are within their spirit and scope.

I claim:

1. A computer network security method, the method comprising the steps of:

monitoring a communications traffic stream of the computer network, the communications traffic stream including a plurality of files;

inserting a probe into at least one file of the plurality of files;

determining whether the probe is executed in the computer network; and

in response to the execution of the probe, identifying a location within the computer network where the execution of the probe occurred.

2. The method of claim **1** further comprising the step of:

generating a security alert containing at least the identified location within the computer network.

3. The method of claim **2** wherein the identified location is a particular user terminal of a plurality of user terminals within the computer network.

4. The method of claim **1** wherein the inserting the probe step occurs in a server within the computer network.

5. The method of claim **2** wherein the probe is a computer virus configured as a trojan horse.

6. The method of claim **4** wherein the communications traffic stream passes through the server as the communications traffic stream is exchanged between the computer network and a public network.

7. The method of claim **3** wherein the execution of the probe occurs in a web browser running on the particular user terminal.

8. The method of claim **5** wherein the security alert is generated as a function of a UDP packet transmitted by the trojan horse.

9. A method for providing security in a private network, the private network having a plurality of user terminals, the method comprising the steps of:

monitoring a communications traffic stream between the private network and a public network, the communications traffic stream including a plurality of files, particular ones of the plurality of files destined for particular ones of the plurality of user terminals;

inserting at least one probe of a plurality of probes into the particular ones of the plurality of files;

determining whether the probe is executed by the particular one of the user terminals for which the file was destined; and

in response to the execution of the probe, identifying the particular one of the user terminals in which the execution of the probe occurred.

10. The method of claim **9** wherein the inserting the at least one probe step occurs in a firewall situated between the private network and the public network.

11. The method of claim **10** comprising the further step of:

transmitting a security alert from the probe to the firewall, the security alert containing an indication of at least the identified user terminal.

12. The method of claim **10** wherein the inserting the at least one probe step occurs as a function of a first access to the public network from at least one user terminal.

13. The method of claim **12** wherein the probe includes at least one Javascript instruction.

14. The method of claim **9** wherein the communications traffic stream comprises a plurality of TCP/IP packets.

15. A method for use in a firewall which provides security between a private network and a public network, the method comprising the steps of:

monitoring a communications traffic stream transmitted between the private network and the public network, the communications traffic stream including a plurality of packets;

inserting a probe into at least one packet of the plurality of packets;

determining whether the probe is executed in the private network; and

in response to the execution of the probe, identifying a location within the private network where the execution of the probe occurred.

16. The method of claim **15** wherein the private network is a computer network having a plurality of user terminals.

17. The method claim **16** wherein the identifying the location step further comprises transmitting a signal from the probe to the firewall indicating that the probe has executed.

18. The method claim **16** wherein the inserting the probe step occurs as a function of a first access to the public network from at least one user terminal.

19. A network security apparatus comprising:

a prober for inserting a plurality of probes into a plurality of packets exchanged between a private network and a public network; and

a processor for monitoring the plurality of packets and determining whether particular ones of the plurality of probes are executed in the private network.

20. The network security apparatus of claim **19** further comprising:

a database for storing the plurality of probes.

21. The network security apparatus of claim **19** further comprising a communications channel for downloading the plurality of probes from a central source.

22. A network security method, the method comprising the steps of:

inserting a plurality of probes into an incoming communications stream of a private network; and

monitoring a plurality of user terminals in the private network for a execution of at least one probe of the plurality of probes.

23. The method of claim **22** further comprising the step of:

generating a report which identifies particular ones of a plurality of user terminals in the private network in which probes have executed.

24. The method of claim **22** wherein the monitoring the plurality of user terminals step further comprises transmitting a signal to a firewall indicating the execution of the at least one probe.

25. The method of claim **24** wherein the inserting the plurality of probes step occurs within a firewall.

26. The method of claim **24** wherein the incoming communications stream is from a public network.

27. The method of claim **26** wherein the inserting the plurality of probes step occurs as a function of a request from the private network for accessing a particular resource within the public network.

28. The method of claim **26** wherein the inserting the plurality of probes step occurs as a function of a first access to the public network from at least one user terminal.

* * * * *

US006577920B1

(12) **United States Patent**
Hyppönen et al.

(10) **Patent No.: US 6,577,920 B1**
(45) **Date of Patent: Jun. 10, 2003**

(54) COMPUTER VIRUS SCREENING

(75) Inventors: **Mikko Hyppönen**, Espoo (FI); **Ari Hyppönen**, Espoo (FI); **Mikko Kuisha**, Helsinki (FI); **Urmas Rahu**, Espoo (FI); **Risto Siilasmaa**, Espoo (FI)

(73) Assignee: **Data Fellows Oyj**, Espoo (FI)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/165,279**

(22) Filed: **Oct. 2, 1998**

(51) **Int. Cl.**[7] **H04L 9/00**

(52) **U.S. Cl.** **700/200**; 713/176; 713/201

(58) **Field of Search** 713/201, 164, 713/188, 200, 176

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,311,591 A	* 5/1994	Fischer	....	380/4
5,475,839 A	* 12/1995	Watson et al.	....	395/650
5,951,698 A	* 9/1999	Chen et al.	....	714/38
5,956,481 A	* 9/1999	Walsh et al.	....	395/186
5,960,170 A	* 9/1999	Chen et al.	....	395/183
5,974,141 A	* 10/1999	Saito	....	380/4
5,978,917 A	* 11/1999	Chi	....	713/201
6,006,329 A	* 12/1999	Chi	....	713/200
6,094,731 A	* 7/2000	Waldin et al.	....	714/38
6,108,799 A	* 8/2000	Boulay et al.	....	713/200

OTHER PUBLICATIONS

Microsoft Press; Microsoft Corporation, Microsoft Press Computer Dictionary. 1997, 3rd Edition, pp. 129,294,302, 327, and 430.*
Microsoft Press: Computer Dictionary: 1997 Microsoft Press, Redmond, Washington. 3rd Edition, p. 294.*
Adam, John A.; Data Security: Cyptography-Privacy?, IEEE Spectrum. Aug. 1992, New York, start p. 29.*
Schneier, Bruce; John Wiley ? Sons, Inc., Applied Cryptography, 2nd Edition. 1996, Canada, pp. 34–41.*
Okamoto, E.; ID-based authentication system for computer virus detection, Electronics Letters. Jul. 1990, Kawasaki, Japan, pp. 1169–1170.*
Omura, J.K.; Novel applications of cryptography in digital communications, IEEE Communications Magazine, May 1990, Los Angeles, CA, pp. 21–29.*
Macro Mania, Northstar Solutions.com. Copyright 1996–2002, 2 pages.*
Commercial Add-ons, Surpac.com, 2 pages.*
Klaming, Guenther; Download Turbo Font, Turbofnt.com. 1999, 2 pages.*
Anonymous; Combat Macro Viruses with Digital Signatures, ZD Journals. Jan. 2000, Louisville, vol. 7, pp. 4–8.*

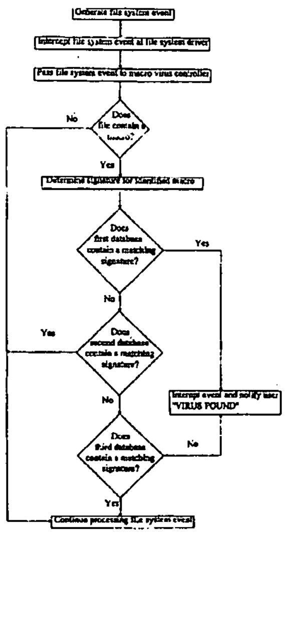
* cited by examiner

Primary Examiner—Gail Hayes
Assistant Examiner—Leynna Ha
(74) *Attorney, Agent, or Firm*—Browdy and Neimark, P.L.L.C.

(57) ABSTRACT

A method of screening a software file for viral infection comprising defining a first database of known macro virus signatures, a second database of known and certified commercial macro signatures, and a third database of known and certified local macro signatures. The file is scanned to determine whether or not the file contains a macro. If the file contains a macro, a signature for the macro is determined and screened against the signatures contained in said databases. A user is alerted in the event that the macro has a signature corresponding to a signature contained in said first database and/or in the event that the macro has a signature which does not correspond to a signature contained in either of the second and third databases.

14 Claims, 2 Drawing Sheets


Generate file system event
Intercept file system event at file system driver
Pass file system event to macro virus controller
Does file contain a macro?
No
Yes
Determine signature for identified macro
Does first database contain a matching signature?
Yes
No
Does second database contain a matching signature?
Yes
No
Interrupt event and notify user "VIRUS FOUND"
Does third database contain a matching signature?
No
Yes
Continue processing file system event

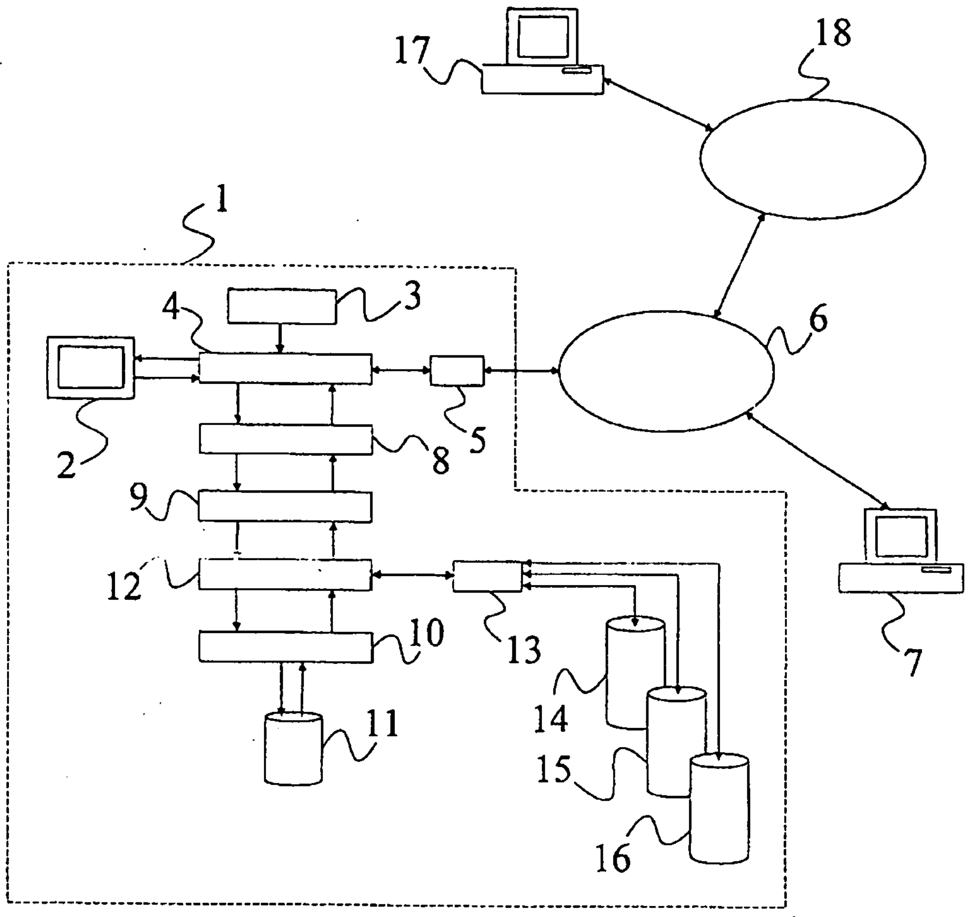


Figure 1

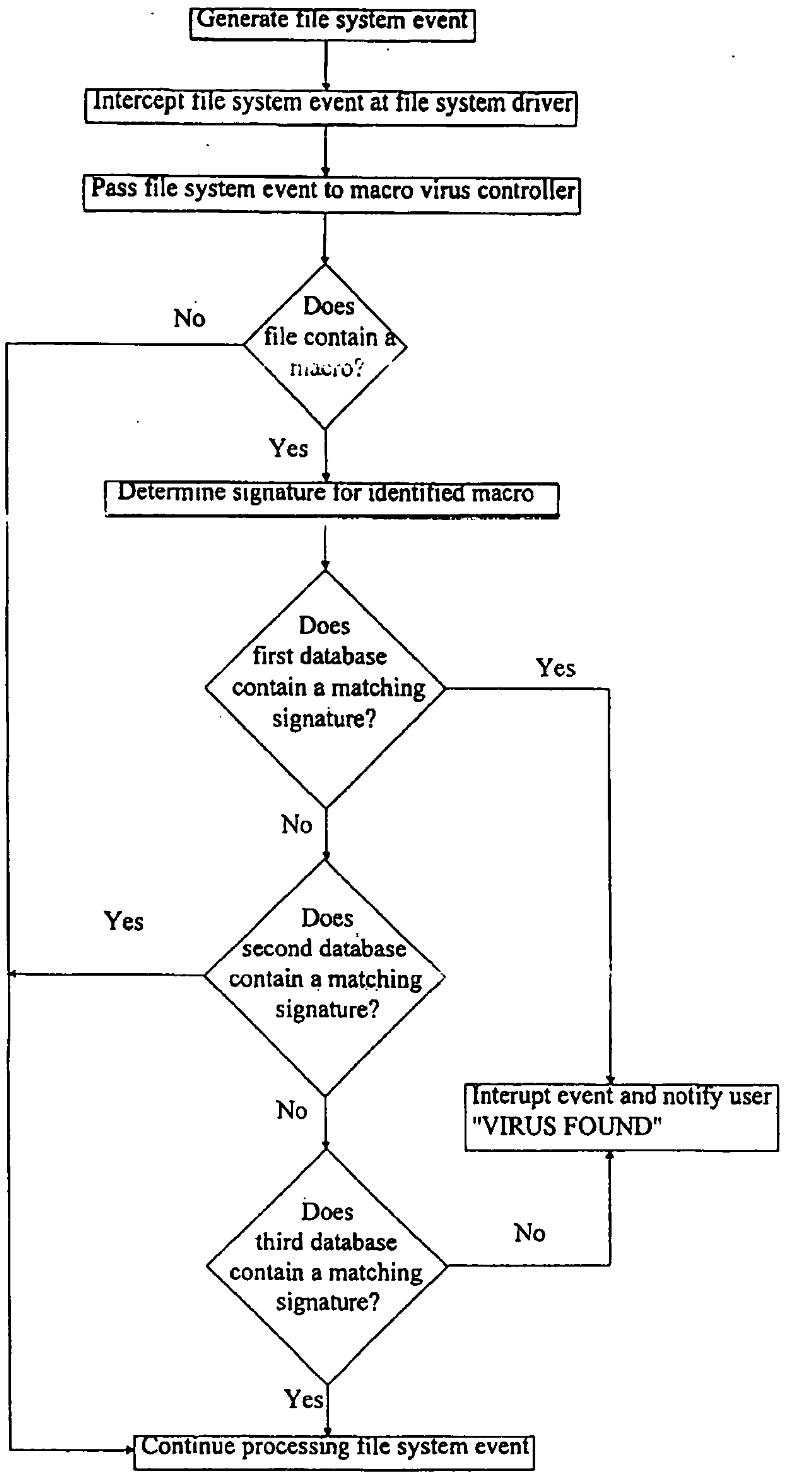

Generate file system event
Intercept file system event at file system driver
Pass file system event to macro virus controller
Does file contain a macro?
No
Yes
Determine signature for identified macro
Does first database contain a matching signature?
Yes
No
Does second database contain a matching signature?
Yes
No
Interupt event and notify user "VIRUS FOUND"
Does third database contain a matching signature?
No
Yes
Continue processing file system event

Figure 2

COMPUTER VIRUS SCREENING

FIELD OF THE INVENTION

The present invention relates to the screening of computer data for viruses and more particularly to the screening of computer data for macro viruses.

BACKGROUND OF THE INVENTION

Computer data viruses represent a potentially serious liability to all computer users and especially to those who regularly transfer data between computers. Computer viruses were first identified in the 1980's, and up until the mid-1990s consisted of a piece of executable code which attached itself to a bona fide computer program. At that time, a virus typically inserted a JUMP instruction into the start of the program which, when the program was executed, caused a jump to occur to the "active" part of the virus. In many cases, the viruses were inert and activation of a virus merely resulted in its being spread to other bona fide programs. In other cases however, activation of a virus could cause malfunctioning of the computer running the program including, in extreme cases, the crashing of the computer and the loss of data.

Computer software intended to detect (and in some cases disinfect) infected programs has in general relied as a first step upon identifying those data files which contain executable code, e.g. .exe, .com, .bat. Once identified, these files are searched (or parsed) for certain signatures which are associated with known viruses. The producers of anti-virus software maintain up to date records of such signatures which may be, for example, checksums.

WO95/12162 describes a virus protection system in which executable data files about to be executed are passed from user computers of a computer network to a central server for virus checking. Checking involves parsing the files for signatures of known viruses as well as for signatures of files known to be clean (or uninfected).

In 1995, a new virus strain was identified which infected, in particular, files of the Microsoft Office™ system. Given the dominant position of Microsoft Office™ in the computer market, the discovery of these viruses has caused much consternation.

Microsoft Office™ makes considerable use of so-called "macros" which are generally small executable programs written in a simple high level language. Macros may be created, for example, to provide customised menu bars or "intelligent" document templates or may be embedded in some other file format. For example, macros may be embedded in template files (.dot) or even in Microsoft Word™ files (.doc).

As the new strains of virus discovered in 1995 infect macro files, they are generally referred to as "macro viruses". It will be appreciated that the possibility for macro viruses to be spread is great given the frequency with which Microsoft Office™ files are copied between two computers either by way of floppy disk or via some other form of electronic data transfer, e.g. the Internet. Indeed, viruses such as "WM/Concept" are known to have spread widely and rapidly at a global level.

Producers of anti-virus software have approached the macro virus problem by maintaining and continuously updating records of macro viruses known to exist in the "wild". As with more conventional viruses, a signature (commonly a checksum) is determined for each macro virus

and these signatures are disseminated to end users of anti-virus software. The software generally scans data being written to or read from a computer's hard disk drive for the presence of macros having a checksum corresponding to one of the identified viruses.

There are a number of problems with these more or less conventional approaches. Firstly, the number of macro viruses is exploding with around 3000 identified by mid 1998. There is inevitably a time lag between a virus being released and its being identified, by which time many computers may have been infected. Secondly, end users may be slow in updating their systems with the latest virus signatures. Again, this leaves a window of opportunity for systems to be infected.

WO 98/14872 describes an anti-virus system which uses a database of known virus signatures as described above, but which additionally seeks to detect unknown viruses based upon expected virus properties. However, given the ingenuity of virus producers, such a system is unlikely to be completely effective against unusual and exotic viruses.

SUMMARY OF THE PRESENT INVENTION

It is an object of the present invention to overcome or at least mitigate the above noted disadvantages of existing anti-virus software.

This and other objects are met by screening computer data to identify macros which do not correspond to known certified and acceptable macros.

According to a first aspect of the present invention there is provided a method of screening a software file for viral infection, the method comprising;

defining a database of signatures indicative of macros previously certified as being virus free;

scanning said file to determine whether or not the file contains a macro; and

if the file contains a macro, determining whether or not the macro has a signature corresponding to one of the signatures contained in said database.

It will be appreciated that embodiments of the present invention have the advantage that they may be used to effectively block the transfer and/or processing of files which contain a previously unidentified (either to the local user or to the software producer) macro virus. It is therefore less critical (or even unnecessary) for the software to be updated to take account of newly detected viruses).

Preferably, said step of defining a database of signatures indicative of macros previously certified as being virus free comprises scanning a set of end user applications which are known to be virus free to identify macros therein, determining a signature for each of the identified macros, and compiling the determined signatures into the database. More preferably, the step of defining the database comprises the further steps of updating the database with additional macro signatures. This updating may be done via an electronic link between a computer hosting the database (where the scanning of the file is performed) and a remote central computer. Alternatively, the database may be updated by way of data stored on an electronic storage medium such as a floppy disk. The database may also include signatures corresponding to widely used proprietary macros, e.g. those used by large organisations.

Preferably, the method comprises defining a second database comprising signatures indicative of macro viruses, and scanning said file to determine whether or not the file contains a signature corresponding to one of signatures contained in the second database. This second database may be created at a central site and disseminated to end users by floppy disk or direct electronic data transfer.

Preferably, the method comprises creating a set of signatures corresponding to a set of user specific macros, certified by the user as being virus free. These signatures may be added to the first mentioned database, or may be included in a separate database. In either case, the method comprises scanning a macro identified in a file to determine whether or not the macro has a signature corresponding to a signature of a user certified macro. The user in this case may be an end user, but preferably is a network manager. In the latter case, database updates made by the network manager are communicated to the network end user computers where the virus screening is performed.

According to a second aspect of the present invention there is provided a method of screening a software file for viral infection, the method comprising:

defining a first database of known macro virus signatures, a second database of known and certified commercial macro signatures, and a third database of known and certified local macro signatures;

scanning said file to determine whether or not the file contains a macro; and, if the file contains a macro

determining a signature for the macro and screening that signature against the signatures contained in said databases; and

alerting a user in the event that the macro has a signature corresponding to a signature contained in said first database and/or in the event that the macro has a signature which does not correspond to a signature contained in either of the second and third databases.

According to a third aspect of the present invention there is provided apparatus for screening a software file for viral infection, the apparatus comprising;

a memory storing a set of signatures indicative of macros previously certified as being virus free; and

a data processor arranged to scan said file to determine whether or not the file contains a macro and, if the file does contain a macro, to determine whether or not the macro has a signature corresponding to one of the signatures contained in said database.

According to a third aspect of the present invention there is provided a computer memory encoded with executable instructions representing a computer program for causing a computer system to:

maintain a database of signatures indicative of macros previously certified as being virus free;

scan data files to determine whether or not the files contains a macro; and

if a file contains a macro, determine whether or not the macro has a signature corresponding to one of the signatures contained in said database.

Preferably, the computer program provides for the updating of said database with additional macro signatures.

Preferably, the computer program causes a second database to be maintained which comprises signatures indicative of macro viruses, and further causes the files to be scanned to determine whether or not they contain a signature corresponding to one of signatures contained in the second database. More preferably, the computer program causes a third database to be maintained which comprises signatures indicative of macros defined locally, e.g. at the level of a local network to which the programmed computer is connected. The computer program causes this third database to be scanned for a match between signatures of a file macro not already matched in the first and second databases, and signatures contained in the third database.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a functional block diagram of a computer system in which is installed macro virus screening software; and

FIG. 2 is a flow chart illustrating the method of operation of the system of FIG. 1.

DETAILED DESCRIPTION OF CERTAIN EMBODIMENTS

For the purpose of illustration, the following example is described with reference to the Microsoft Windows™ series of operating systems, although it will be appreciated that the invention is also applicable to other operating systems such as Macintosh system and OS/2. With reference to FIG. 1, an end user computer 1 has a display 2 and a keyboard 3. The computer 1 additionally has a processing unit and a memory which provide (in functional terms) a graphical user interface layer 4 which provides data to the display 2 and receives data from the keyboard 3. The graphical user interface layer 4 is able to communicate with other computers via a network interface 5 and a network 6. The network is controlled by a network manager 7.

Beneath the graphical user interface layer 4, a number of user applications are run by the processing unit. In FIG. 1, only a single application 8 is illustrated and may be, for example, Microsoft Word™. The application 8 communicates with a file system 9 which forms part of the Microsoft Windows™ operating system and which is arranged to handle file access requests generated by the application 8. These access requests include file open requests, file save requests, file copy requests, etc. The lowermost layer of the operating system is the disk controller driver 10 which communicates with and controls the computer's hard disk drive 11. The disk controller driver 10 also forms part of the Microsoft Windows™ operating system.

Located between the file system 9 and the disk controller driver 10 is a file system driver 12 which intercepts file system events generated by the file system 9. The role of the file system driver 12 is to co-ordinate virus screening operations for data being written to, or read from, the hard disk drive 11. A suitable file system driver 12 is, for example, the GATEKEEPER™ driver which forms part of the F-SECURE ANTI-VIRUS™ system available from Data Fellows Oy (Helsinki, Finland). In dependence upon certain screening operations to be described below, the file system driver 12 enables file system events to proceed normally or prevents file system events and issues appropriate alert messages to the file system 9.

The file system driver 12 is functionally connected to a macro virus controller 13, such that file system events received by the file system driver 12 are relayed to the macro virus controller 13. The macro virus controller is associated with three databases 14 to 16 which each contain a set of "signatures" previously determined for respective macros. For the purposes of this example, the signature used is a checksum derived using a suitable checksum calculation algorithm, such as the US Department of Defence Secure Hash Algorithm (SHA) or the older CRC 32 algorithm.

The first database 14 contains a set of signatures derived for known macro viruses. The signatures in this database 11 are determined by the provider of the file driver system 12 and the macro virus controller 13 and are regularly updated to take into account newly discovered viruses. Updates may be provided by way of floppy disks or directly by downloading them from a remote server 17 connected to the Internet 18.

The second database 15 contains a set of signatures derived for commercially available macros. These macros include those supplied with the Microsoft Office™ operating system and with user applications such as Microsoft Word™. Again, these signatures are determined by the provider of the file driver system 12 and the macro virus controller 13 and are regularly updated to take into account newly available products.

The third database 16 contains a set of signatures which are derived for macros created and used at the local network level, for example letter templates and the like (of course this database may be empty if no local macros are defined). Once a new local macro is created, typically at the network manager 7, the macro is processed by the network manager 7 to derive the corresponding (checksum) signature. This is then relayed via the local network 6 to the end user computer 1 where it is added to the third database 16. It is usually the case that only the network manager has the authority to modify this database 16, whilst the first and second databases 14,15 can be updated only by the network manager 7 using signatures specified by the anti-virus software provider.

Upon receipt of a file system event, the macro virus controller 13 first analyses the file associated with the event (and which is intended to be written to the hard disk drive 11, read, copied, etc) to determine if the file contains a macro. This may include examining the file name extension (e.g. to identify dot, .doc files) and/or scanning the file for embedded macros. If one or more macros is identified in the file, a checksum signature is determined for the/or each identified macro.

Assuming that a single macro is identified in the file, the macro virus controller 13 scans the first database 14 to determine whether or not the corresponding signature is present in that database 14. If the signature is found there, the macro virus controller 13 reports this to the file system driver 12. The file system driver 12 in turn causes the system event to be suspended and causes an alert to be displayed to the user that a known virus is present in the file. The file system driver 12 may also cause a report to be sent to the network manager 7 via the local network 6.

If this first scan does not locate a known virus, the macro virus controller 13 proceeds to search the second database 15 to determine whether or not the signature for the identified macro is present in that database 15. If the signature is found, then an appropriate report is sent to the file system driver 12, which in turn allows the file event to proceed normally. However, if the signature is not found in the second database 15, this indicates that the identified macro is unknown to the system and may be a new and unknown virus.

Before a warning is issued to the user, the macro virus controller 13 searches the third database 16 to determine whether the as yet unidentified macro corresponds to a locally defined macro. If the answer is yes, then the macro virus controller 13 reports accordingly to the file system driver 12 and the event is allowed to proceed. On the other hand, if the identified macro signature is not found in the third database 16, then the macro virus controller 13 reports this to the file system driver 12 and the event is suspended. Again, a report is sent to the network manager 7, and also possibly to the remote server 17 of the software provider. This report may be accompanied by a copy of the "guilty" macro.

The file scanning system described above is further illustrated by reference to the flow chart of FIG. 2.

It will be appreciated by the person of skill in the art that various modifications may be made to the embodiment described above without departing from the scope of the present invention. For example, the file system driver 12 may make use of further virus controllers including controllers arranged to screen files for viruses other than macro viruses. The file system driver 12 may also employ disinfection systems and data encryption systems.

It will also be appreciated that the file system driver 12 typically receives all file access traffic, and not only that relating to hard disk access. All access requests may be passed to the macro virus controller 13 which may select only hard disk access requests for further processing or may also process other requests relating to, but not limited to, floppy disk data transfers, network data transfers, and CDROM data transfers.

We claim:

1. A method of screening a software file for viral infection, the method comprising:

defining a first database of known macro virus signatures, a second database of known and certified commercial macro signatures, and a third database of known and certified local macro signatures;

scanning said file to determine whether or not the file contains a macro; and, if the file contains a macro

determining a signature for the macro and screening that signature against the signatures contained in said databases; and

alerting a user in the event that the macro has a signature corresponding to a signature contained in said first database and/or in the event that the macro has a signature which does not correspond to a signature contained in either of the second and third databases.

2. A method according to claim 1, wherein said step of defining a second database of known and certifiable commercial macro signatures comprises scanning a set of end user applications which are known to be virus free to identify macros therein, determining a signature for each of the identified macros, and compiling the determined signatures into the second database.

3. A method according to claim 1, wherein the step of defining the third database comprises the further steps of updating the third database with additional macro signatures.

4. A method according to claim 3, wherein said updating steps are done via an electronic link between a computer hosting the database, where the scanning of the file is performed, and a remote central computer.

5. A method according to claim 1, wherein thee user is a network manager and database updates made by the network manager are communicated to network end user computers where virus screening is performed.

6. A method according to claim 1, wherein said step of determining a signature for the macro and screening that signature comprises deriving a signature of the macro and comparing the derived signature with signatures in the databases.

7. A method of screening a software file to determine whether any macro contained therein does or does not contain a virus, the method comprising:

defining a first database of known macro virus signatures, a second database of known and certified commercial macro signatures, and a third database of known and certified local macro signatures;

scanning said file to determine whether or not the file contains a macro; and if the file contains a macro, determining whether or not the macro has a signature corresponding to one of the signatures contained in said databases.

8. Apparatus for screening a software file for viral infection, the apparatus comprising:

a memory storing a first database of known macro virus signatures, a second database of known and certified commercial macro signatures, and a third database of known and certified local macro signatures; and

a data processor arranged to scan said file to determine whether or not the file contains a macro and, if the file does contain a macro, to determine whether or not the macro has a signature corresponding to one of the signatures contained in said databases.

9. The apparatus according to claim 8, wherein, in order to determine whether or not the macro has a signature corresponding to one of the signatures contained in said databases, said data processor is arranged to derive a signature of the macro and to compare the derived signature with signatures in the databases.

10. A computer memory encoded with executable instructions representing a computer program for causing computer system to:

maintain a first database of known macro virus signatures, a second database of known and certified commercial macro signatures, and a third database of known and certified local macro signatures;

scan data files to determine whether or not the files contains a macro; and

if a file contains a macro, determine whether or not the macro has a signature corresponding to one of the signatures contained in said second database.

11. A computer memory according to claim 10, wherein the computer program provides for the updating of said third database with additional macro signatures.

12. A computer memory according to claim 10, wherein the computer program causes the files to be scanned to determine whether or not they contain a signature corresponding to one of signatures contained in the first database.

13. A computer memory according to claim 12, wherein the computer program causes the third database to be scanned for a match between signatures of a file macro not already matched in the first and second databases, and signatures contained in the third database.

14. The computer memory according to claim 10, wherein in order to determine whether or not the macro has a signature corresponding to one of the signatures contained in said databases, said computer program causes the computer system to derive a signature of the macro and to compare the derived signature with signatures in the databases.

* * * * *

L Number	Hits	Search Text	DB	Time stamp
1	2	("6006242").PN.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/13 07:46
2	2	("6052718").PN.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/13 07:49
3	12	version and (timestamp with mask)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/13 07:50
4	43	version and (signature with mask)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/13 07:50
5	55	(version and (timestamp with mask)) or (version and (signature with mask))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/13 07:50
6	4	((version and (timestamp with mask)) or (version and (signature with mask))) and virus	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/13 07:51
7	28	((version and (timestamp with mask)) or (version and (signature with mask))) and update	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/13 07:51
8	20	(((version and (timestamp with mask)) or (version and (signature with mask))) and update) and event	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/13 07:52
9	14	(signature with policy) and "file system" and virus	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/13 07:54
10	7	(mask with (virus or corrupt$)) and "file system"	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/13 07:58
11	833	(707/203).ccls.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/13 07:58
12	1	((707/203).ccls.) and (version and (signature with mask))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/13 07:58

L Number	Hits	Search Text	DB	Time stamp
1	2	("6192340").PN.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/05/14 10:27
2	2	("6204441").PN.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/05/14 10:27
3	2	("6415277").PN.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/05/14 11:31
4	2	("6226561").PN.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/05/14 11:32
-	96	(mark with information) and (meta with information)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/05/14 10:06
-	13	((mark with information) and (meta with information)) and signature	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/05/13 16:37
-	0	(((mark with information) and (meta with information)) and signature) and virus	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/05/13 16:37
-	5	(((mark with information) and (meta with information)) and signature) and corrupt$	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/05/13 16:37
-	6	(((mark with information) and (meta with information)) and signature) and scan	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/05/13 16:38
-	35	(mask with information) and (meta with information)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/05/13 16:42
-	11	((mask with information) and (meta with information)) and signature	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/05/13 16:45
-	9	((mask with information) and (meta with information)) and signature and (virus or corrupt$)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/05/13 16:45
-	2	((mask with information) and (meta with information)) and signature and (virus)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/05/13 16:46
-	213	signature with policy	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/05/13 16:47

L Number	Hits	Search Text	DB	Time stamp
1	274	(signature or license) and "file system" and cop$2 and virus and update	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/13 08:11
2	1	((signature or license) and "file system" and cop$2 and virus and update) and (signature with policy)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/13 08:13
3	3	(virus with scan) and (signature with policy)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/13 08:13
4	137	(virus with scan) and (signature)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/13 08:13
5	56	((virus with scan) and (signature)) and "file system"	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/13 08:14
6	12	(((virus with scan) and (signature)) and "file system") and policy	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/06/13 08:14

> home > about > feedback > login
US Patent & Trademark Office

Try the *new* Portal design
Give us your opinion after using it.

Search Results

Search Results for: **[meta and mask <AND>((version and file system and virus and network))]**
Found **1** of **111,041 searched.** → Rerun within the Portal

Search within Results

> Advanced Search
> Search Help/Tips

Sort by: Title Publication Publication Date Score Binder

Results 1 - 1 of 1 short listing

1 Practical byzantine fault tolerance and proactive recovery 77%
Miguel Castro , Barbara Liskov
ACM Transactions on Computer Systems (TOCS) November 2002
Volume 20 Issue 4
Our growing reliance on online services accessible on the Internet demands highly available systems that provide correct service without interruptions. Software bugs, operator mistakes, and malicious attacks are a major cause of service interruptions and they can cause arbitrary behavior, that is, Byzantine faults. This article describes a new replication algorithm, BFT, that can be used to build highly available systems that tolerate Byzantine faults. BFT can be used in practice to implement re ...

Results 1 - 1 of 1 short listing

PORTAL
ACM DIGITAL LIBR

> home : > about : > feedback : > login
US Patent & Trademark Office

Try the *new* Portal design
Give us your opinion after using it.

Search Results

Search Results for: **[corrupt and scan<AND>((security <AND>((digital signature and policy<AND>((version and file system and virus and network))))))]**
Found **2** of **111,041 searched.** → Rerun within the Portal

Search within Results

> Advanced Search
> Search Help/Tips

Sort by: Title Publication Publication Date Score Binder

Results 1 - 2 of 2 short listing

1 Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining — 77%
Robbert Van Renesse , Kenneth P. Birman , Werner Vogels
ACM Transactions on Computer Systems (TOCS) May 2003
Volume 21 Issue 2
Scalable management and self-organizational capabilities are emerging as central requirements for a generation of large-scale, highly dynamic, distributed applications. We have developed an entirely new distributed information management system called Astrolabe. Astrolabe collects large-scale system state, permitting rapid updates and providing on-the-fly attribute aggregation. This latter capability permits an application to locate a resource, and also offers a scalable way to track sys ...

2 Bidirectional mobile code trust management using tamper resistant hardware — 77%
John Zachary , Richard Brooks
Mobile Networks and Applications April 2003
Volume 8 Issue 2
Trust management in a networked environment consists of authentication and integrity checking. In a mobile computing environment, both remote hosts and mobile code are suspect. We present a model that addresses trust negotiation between the remote host and the mobile code simultaneously. Our model uses tamper resistant hardware, public key cryptography, and one-way hash functions.

Results 1 - 2 of 2 short listing